Fundamentals of Computer and Programming

Lecture 5
Interaction

Instructor: Morteza Zakeri, Ph.D. (m-zakeri@live.com)

Modified Slides from Dr. *Hossein Zeinali* and *Dr. Bahador Bakhshi*

Computer Engineering Department, Amirkabir University of Technology

Spring 2025

Interaction

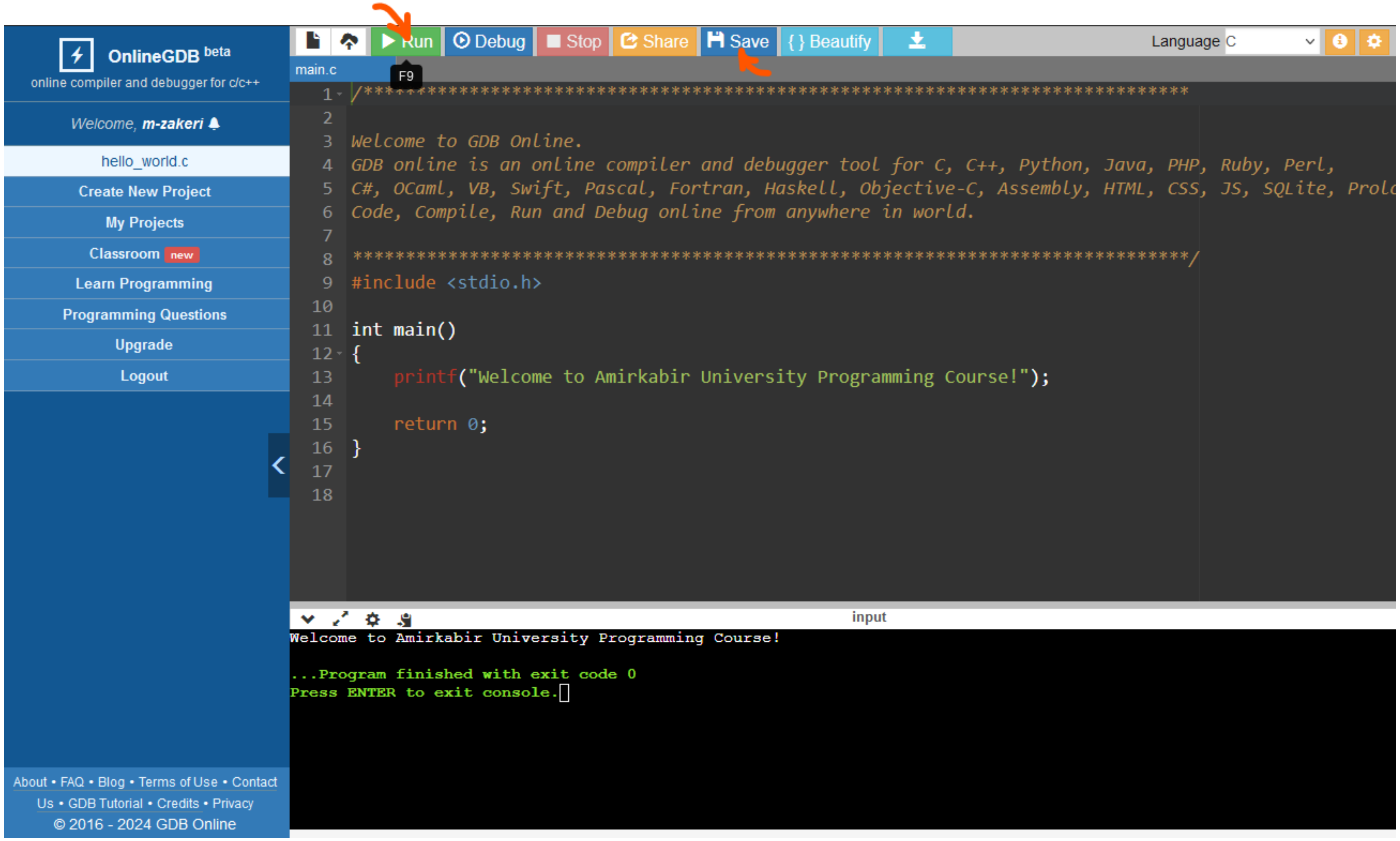
- Produce output
- Get input values

Different kinds of interactions

- **Input:** Directly from keyboard, Mouse in GUI, Microphone, Joystick, …
- **Output:** Directly message on screen, Windows in GUI, Sound card, …
- In this course we use the simple method
  - directly read from the keyboard and write to the screen
  - called "**Console**" **or** " **Terminal**"
- In Graphical OS (like Windows), the console is simulated by OS in a window

C Online Compilers

- https://www.onlinegdb.com/online_c_compiler



```c
/******************************************************************************

Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,
C#, OCaml, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prol
Code, Compile, Run and Debug online from anywhere in world.

*******************************************************************************/
#include <stdio.h>

int main()
{
    printf("Welcome to Amirkabir University Programming Course!");

    return 0;
}
```

```
Welcome to Amirkabir University Programming Course!

...Program finished with exit code 0
Press ENTER to exit console.
```

Interaction

- Produce output
- Get input values

Printing

- Printing messages

```
printf("This is message \n");
//‘\n’ prints a new line
```

- Printing variables
  - `printf("format specifier", parameters);`
  - `format specifier = %[flags][width][.precision]specifier`

```
int i = 20;
char c = 'a';
printf("%d, %c", i, c);
printf("i is %d and char is %c", i, '6');
```

Printing Integers

- `%d, %i, %ld`
  - `%i` is the same as `%d` `in printf`

```
printf("%d", 100);
// 100
printf("%d, %d", +1000, -100);
// 1000, -100
printf("%i", 100);
// 100
printf("%ld, %i", +1000, -100);
// 1000, -100
```

Printing Unsigned Integers

- **%u** (base 10), **%o** (base 8), **%x** (base 16) and **%X** (Base 16)

```
unsigned int i = 26;
printf("%u\n", i);              //26
printf("%o\n", i);              //32
printf("%x\n", i);              //1a
printf("%X\n", i);              //1A
```

Printing Floats

- **%f, %e, %E, %lf**

```
printf("%f", 100.5f);
// 100.500000

float f = -2;
double d = 100;
printf("%f, %lf", f, d);
// -2.000000, 100.000000

printf("%f, %e", 1e3, 1e3);
// 1000.000000, 1.000000e+003
```

Printing Chars

- **%c**

```
printf("%c", 'a’); // a


printf("%c, %c", 'a','b’); // a, b

char c1 = 'a';
printf("%c, %c, %c", c1, 'b', 65); // a, b, A
```

Special Character

➢ Characters in **`printf`**	The result
\n	newline
\t	tab
\r	carriage return
\b	backspace
\"	"
\%	%
%%	%

Printing Strings

- %s

```
printf("This is message");
// This is message

printf("This is %s", "message");
// This is message


char str1[20] = "This is message";
printf("%s", str1);
// This is message
```

Field length (width)

- Field length is a number
- Comes after % (and before the format specifier)
- It is the minimum space reserved for print
  - If value is smaller than the space
    - Empty space
  - If value is larger than the space
    - No effect

Field length

```
printf("|%4d|\n", 1);        // |   1|
printf("|%4d|\n", 12345);          // |12345|
printf("|%4d|\n", -12345);         // |-12345|
printf("|%4f|\n", 1234.0f);  // |1234.000000|
printf("|%15f|\n", 1234.0f); // |    1234.000000|
printf("|%4c|\n", 'A');            // |   A|
printf("|%-4c|\n", 'A');           // |A   |
printf("|%4s|\n", "ABC");          // | ABC|
printf("|%4s|\n", "ABCDE");  // |ABCDE|
printf("|%6d|\n", 1234);           // |  1234|
printf("|%-6d|\n", 1234);          // |1234  |
```

Precision

- Precision is a .number and comes after %
- For Integer
  - The minimum number of digits
    - If (# of digits < precision) → empty space: Zero's (0)
- For floats
  - With %f, %e
    - The number of digits after .
- For strings
  - The maximum number of characters

Precision

```
printf("|%.4d|\n", 1);          // |0001|
printf("|%.4d|\n", 12345);            // |12345|
printf("|%.4d|\n", -12345);     // |-12345|
printf("|%.4f|\n", 1234.0f);    // |1234.0000|
printf("|%.8f|\n", 234.0f);     // |234.00000000|
printf("|%.4s|\n", "ABC");            // |ABC|
printf("|%.4s|\n", "ABCDEF");         // |ABCD|
```

Field length and Precision

- This is a number with format a.b
  - Comes after %
- First .b determines the .precision
- Then a specifies the field length (width)

Field length and Precision

```
printf("|%10.5d|\n", 12);
// |     00012|
printf("|%3.5d|\n", 12);
// |00012|
printf("|%10.5lf|\n", 1.234567890123);
// |   1.23457|
printf("|%0.5lf|\n", 1.234567890123);
// |1.23457|
printf("|%15.10s|\n", "Hello, world");
// |     Hello, wor|
printf("|%5.10s|\n", "Hello, world");
// |Hello, wor|
```

Variable Field Length & Precision : *

- * can be used to specify field length and precision which is replaced by a variable

```
int i = 30;
int j = 2;
float f = 1.23456789;
printf("%0*.*f\n", i, j, f);

// 00000000000000000000000001.23
```

Cast in printing (do NOT use)

```
int i = -60;
unsigned int j = 4147482648;
float f = -700.05;

printf("i = %u\n", i);
// i = 4294967236

printf("j = %d\n", j);
// j = -147484648


printf("i = %f\n", i); // error in some compilers
// i = 0.000000

printf("f = %d\n", f); // error in some compilers
// f = 1610612736
```

Interaction

- Produce output
- Get input values

Reading

- Read from keyboard (console)
- What should be determined in reading
  - Keyboard enters "characters", so, how to read int, char, …?
    - Which type the chars should be converted?
  - Where should be saved?
- **scanf**("format specifier", *parameters*)
  - Format: The type that input should be converted to
  - Parameters: Where should be saved
- scanf blocks until 'Enter' at the end of input (why?!)
- Reads from beginning until to white spaces (except reading chars)

Reading Integers (base 10)

- **%d, %u, %ld, %lu**

```
int i;
unsigned int j;
long int l;

scanf("%d", &i);
scanf("%u", &j);
scanf("%ld",&l);

-90            → -90 is saved in memory location i
78             → 78 is saved in memory location j
60L            → 60 is saved in memory location l
Spaces at the beginning are ignored
```

Reading Integers (cont'd)

- `%o, %x, %X, %i`

```
scanf("%o", &i);
```

Input: **12** → i = 10

```
scanf("%x", &i);
```

Input: **1a** → i = 26

```
scanf("%i", &i);
```

Input: **12** → i = 12

Input: **012** → i = 10 (It reads in base 8)

Input: **0x12** → i = 18 (It reads in base 16)

Reading floats and doubles

- %f, %lf, %e

```
float f;
double d;
scanf("%f", &f);
scanf("%lf", &d);
90.9            → 90.9 is saved in memory f
88.123456789    → 88.123456789 saved in
                  memory d
```

Spaces at the beginning are ignored

Reading floats and doubles

```
float f1, f2;

scanf("%f", &f1);

scanf("%e", &f2);
```

Input:

```
1.23          → f1 = 1.23
4.56          → f2 = 4.56
```

Input:

```
1.23e+1       → f1 = 12.3
4.56e-1       → f2 = 0.456
```

Reading chars

- **%c**

```
char c1, c2, c3;

scanf("%c", &c1);   /* spaces */
scanf("%c", &c2);
scanf("%c", &c3);
```

Input: `azb →`

```
            c1 = 'a'
            c2 = 'z'
            c3 = 'b'
```

Spaces at the beginning are NOT ignored

Reading chars (cont'd)

- White spaces (space, tab, enter) are **not** ignored when reading char
- To ignore white spaces, use " " before %c

```
scanf("%d%c%d", &i, &c, &j);
```

Input: **123 45** → I = 123 c = ' ' j = 45

```
scanf("%d %c%d", &i, &c, &j);
```

Input: **123 4 56** → I = 123 c = '4' j = 56

Input: **123 456** → I = 123 c = '4' j = 56

Reading chars (cont'd)

- **`getchar()`**
  - Read char after Enter
- **`getch()`**
  - Read char without Enter, does NOT show the char
    - A non-standard function declared in "**conio.h**" header file.
    - Mostly it is used by Turbo C.
    - It is not a part of C standard library.
- **`getche()`**
  - Read char without Enter, shows the char

Reading Strings

- %s

```
char str[20]; // Defines string with len 20
scanf("%s", str);
```

Input: `ABC` → `str = "ABC"`

```
scanf("%s", str);
```

Input: `AB C` → `str = "AB"`

Reading Strings

- How to read a line
  - Contains spaces (read until end of line)
- `gets(s)`

```
char str[20];
gets(str);
```

Input: `ABC DEF` → `str = "ABC DEF"`

Field length in scanf

- Field length specifies the maximum number of input characters (in the buffer) used for scanning

```
int i, j;
scanf("%5d", &i);
```

Input: 122 → i = 122

Input: 1234567 → i = 12345

```
scanf("%5d%d", &i, &j);
```

Input: 1 2 → i = 1, j = 2

Input: 1234567 → i = 12345, j = 67

Input: 123456 7 → i = 12345, j = 6

Special input format

- If input data has special format with extra characters
  - scanf can ignore them

```
int sal, mah, rooz;
scanf("%d/%d/%d", &sal, &mah, &rooz);
```

Input: `1389/12/1`

→

```
sal = 1389, mah = 12, rooz = 1
```

Format of actual input data

- The format of actual input data **MUST** match with the format of **`scanf`**

```
int a, b;

float f;

scanf("%d--%d%f", &a, &b, &f);
```

Input: `1--2 3.0` → `a = 1, b = 2, f = 3.0`

Input: `1-2 3.0` → `a = 1, b, f without change`

Input: `1.0--2 3.0` → `a = 1, b, f without change`

Common bugs

- Casting in **`printf`** or **`scanf`**
  - `printf("%d", 120.23);`
  - `double d; scanf("%f", &d);`
- Mismatch between format and the number of expressions
  - `printf("%d %d", 10);`
  - `printf("%d", 10, 20);`
- Using name of variable instead of address
  - `scanf("%d", i);`

A running example

```
#include <stdio.h>
#include <stdlib.h>
int main(void){
   int i;
   unsigned int j;
   unsigned long int k;
   char c;
   float f;
   printf("Enter a char:\n");
   scanf(" %c", &c);
   printf("Enter an int:\n");
   scanf("%d", &i);
   printf("Enter an unsigned int:\n");
   scanf("%u", &j);
   printf("Enter an unsigned long int:\n");
   scanf("%lu", &k);
   printf("Enter a float:\n");
   scanf("%f", &f);
```

برنامه‌ای که با تولید پیغام‌های مناسب ورودی‌های را از کاربر بگیرد و در انتها لیست ورودی‌ها را به کاربر نشان دهد.

A running example (cont'd)

```
    printf("Your input are:\n");
    printf("int = %d, unsigned int = %u, unsigned long int = %lu,
       ", i, j, k);

    printf("char = %c and float = %f\n", c, f);

    return 0;
}
```

Quiz

- **Q1:** Write a program to read three scores, their weights, and compute the weighted average of the scores.

- **Q2:** Write a C program that convert a temperature from *Centigrade* to *Fahrenheit.*
  - C = (5/9) * (F - 32)

Equation :

$$\frac{C}{5} = \frac{F - 32}{9}$$

Reference

- Reading Assignment: Chapter 9 of "C How to Program"
- Many programming problems with solutions:
  - *https://m-zakeri.github.io/CP/problems/*

Questions

- How would you modify scanf to read a date in the format YYYY/MM/DD into three integer variables?

  A) scanf("%d/%d/%d", &year, &month, &day);

  B) scanf("%d-%d-%d", &year, &month, &day);

  C) scanf("%d%d%d", &year, &month, &day);

  D) scanf("%s", date);

- Answer: A

Questions

- What will the following code output?

```
int width = 6, precision = 2;
float value = 12.3456;
printf("|%*.*f|\n", width, precision, value);
```

A) | 12.35|

B) | 12.34|

C) |12.35|

D) Compiler error

- Answer: A